AKAMAI THREAT ADVISORY

Threat Advisory: Mirai Botnet

Risk Factor: Medium
TLP: Green

Author: Chad Seaman

Akamai
FASTER FORWARD

1.0 / Overview / Much is already known about the Mirai botnet, due to a thorough write-up by Malware Must Die as well as a later publicly distributed source-code repository. This advisory provides information about attack events and findings prior to the Mirai code release as well as those occurring following its release. The advisory will also summarize pertinent research data and ultimately the processes that led to the associated findings. Signatures observed in real-world attacks are also included and may aid in the future detection and mitigation of Mirai-based attacks.

2.0 / Attack Events Timeline, Statistics, & Signatures / Mirai attack signatures were first observed in attacks against a security blog run by journalist Brian Krebs. The first attack, in the series of four, peaked at 623 Gbps. The timeline below represents the four attacks mitigated by Akamai.

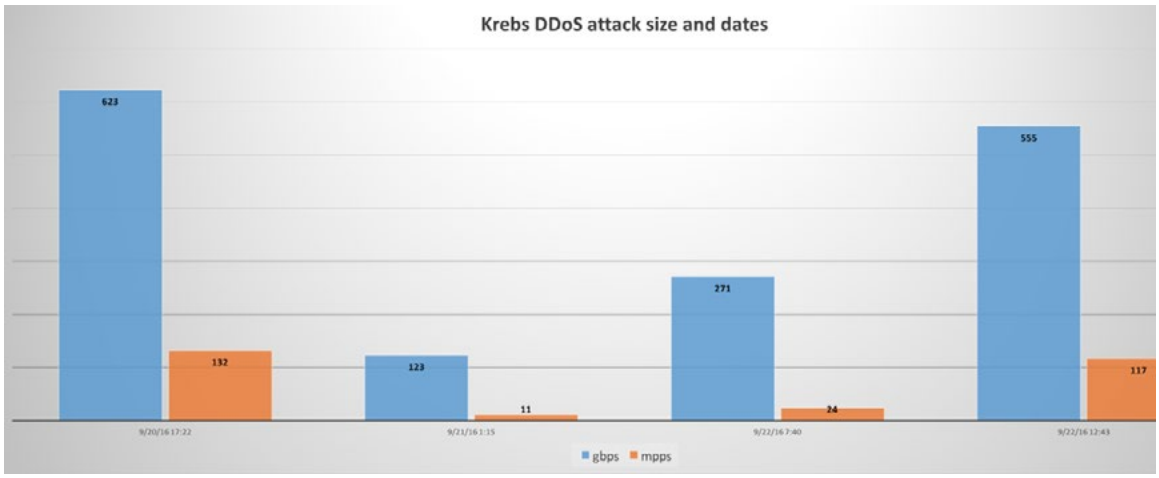

Figure 1: First series of observed Mirai attacks launched against Brian Krebs.

Just days after this series of DDoS attacks, the source code for Mirai was made public. The next timeline represents the bandwidth in gigabits per second for Mirai-confirmed attacks occurring after this code was released. The bandwidth peak, although still substantial, has been observed at mostly under 100 Gbps in later attacks. In addition, most of the attacks were under 30 million packets per second.

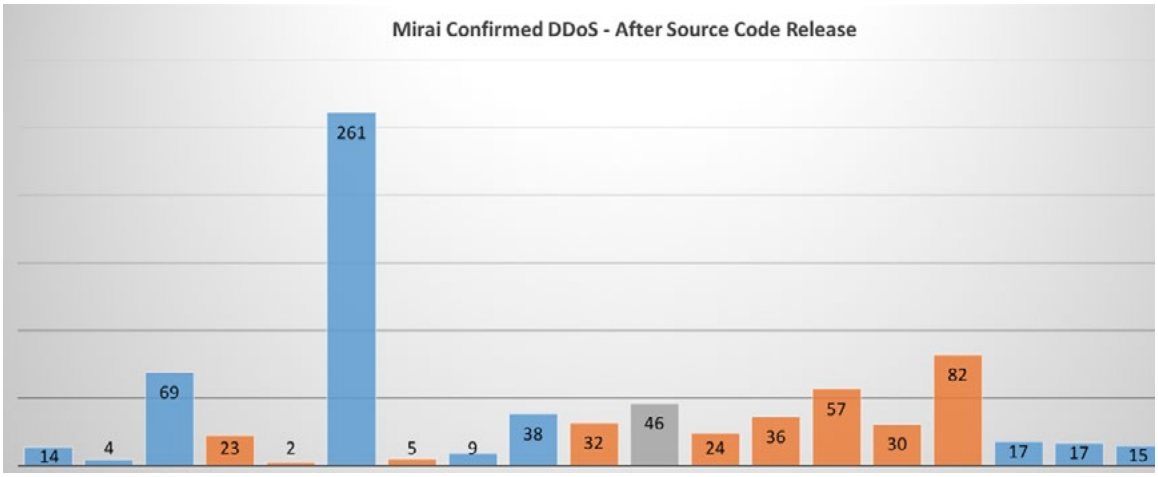

Figure 2: Timeline of attacks that Akamai mitigated following the Mirai code release

The only attack peaking at just over the 30 million packet-per-second mark was the 261 Gbps attack on October 11. The overall lower packet rates can be attributed for the most part to the extra padding in many of the Mirai attacks seen so far. Most of these attack events used vectors with payloads padded with at least 512 bytes of data. These larger packets, while able to consume more bandwidth, typically have a lower packet throughput.

Altogether, the following signatures were observed during Mirai attacks mitigated by Akamai.

These may also provide an indication of common attack vectors.

```
SYN Flood
17:22:33.651224 IP 177.21.60.105.47638 > x.x.x.x.443: Flags [S], seq 2014760206,
win 0, options [mss 1360,sackOK,TS val 269068533 ecr 0,nop,wscale 6], length 0
17:22:33.651247 IP 113.174.205.202.14778 > x.x.x.x.443: Flags [S], seq 3023184484,
win 0, options [mss 1398,sackOK,TS val 2640952386 ecr 0,nop,wscale 6], length 0

GET Flood
20:32:33.549360 IP 60.148.127.231.43862 > x.x.x.x.80: Flags [P.], seq
4294961474:4294961856, ack 2761, win 0, length 382: HTTP: GET /2016/09/ddos-
mitigation-firm-has-history-of-hijacks/ HTTP/1.1
....E.....@.;..&<...H4...V.P3jIJ....P...}>..GET /2016/09/ddos-mitiga-
tion-firm-has-history-of-hijacks/ HTTP/1.1
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/52.0.2743.116 Safari/537.36
Host: krebsonsecurity.com,krebsonsecurity.com
Connection: keep-alive
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.8


--
```

GRE Protocol Flood

```
10:36:19.210974 IP 118.121.6.131 > x.x.x.x: GREv0, length 544: IP
121.169.140.53.1762 > 131.181.106.162.80: UDP, length 512
10:36:19.210998 IP 116.55.170.129 > x.x.x.x: GREv0, length 544: IP
69.214.245.142.4653 > 101.206.249.207.80: UDP, length 512
10:40:56.691708 IP (tos 0x0, ttl 51, id 37540, offset 0, flags [DF], proto GRE (47),
length 564)
   117.36.155.234 > x.x.x.x: GREv0, Flags [none], length 544
   IP (tos 0x0, ttl 64, id 21855, offset 0, flags [DF], proto UDP (17), length 540)
   234.154.146.42.44058 > 106.232.12.53.80: [udp sum ok] UDP, length 512
.(......h%.El#.....7.Bv........Fw....Tu..}....P..:.2.@....d6.@;..\
B...:.K...`....HH?..Z..........,...z.].p....!.K....'.-]:-.>b.tg.e<.8CC3....C}
d..#......f...'.8Ci.|...@].v..#.ca./...B......k..'.U.&:......S.OCh...K..G.B.sX-
....1>...,.U..jB.e....wp(..P|....l...AOr..5C..Q...Pz...`..l..... @..../%.....MB..

20:32:58.682128 IP 170.0.48.158 > x.x.x.x: GREv0, length 932: IP
160.251.95.134.28589 > 156.92.100.10.24315: UDP, length 900
20:32:58.682982 IP 187.49.236.66 > x.x.x.x: GREv0, length 932: IP
93.199.228.142.32455 > 174.180.222.189.42247: UDP, length 900
```

ACK Flood

```
21:23:42.523050 IP 188.164.136.59.20259 > x.x.x.x.51578: Flags [.], seq
3349565526:3349566038, ack 2424917850, win 22465, length 512
21:23:42.523056 IP 95.67.110.59.395 > x.x.x.x.35195: Flags [.], seq
1218844281:1218844793, ack 3689642006, win 62452, length 512

10:01:31.173253 IP 45.3.30.40.16556 > x.x.x.x.43651: Flags [.], seq
1548267287:1548267799, ack 650890440, win 46660, length 512
.e..E..(....6.
..Z..I..Bd}...r.w.!......Cn.U.|..D.Q#n.......G....x...8.q.......U,!.Z...........
ig...a.+N..|x.&...d..]/*b....
yR$O.../R.6aKC.>..x.....+.s...?j.pe......q.....V.3D.A.....g.,@.._...
{XS._..R..<.=.3a...:..UT..N.o..5.1.cI.|.m..FR..rK..\?....aq}..
```

UDP Flood(VSE)

```
10:00:27.953695 IP 197.1.196.112.42706 > x.x.x.x40.27015: UDP, length 25
10:00:27.953699 IP 188.243.40.113.52919 > x.x.x.x40.27015: UDP, length 25
10:00:54.032442 IP 187.56.139.203.13891 > x.x.x.x40.27015: UDP, length 25
....E..544..6..Z.8......6Ci..!......TSource Engine Query.
```

UDP Flood

```
03:33:02.543594 IP 191.205.101.86.10000 > x.x.x.x.33000: UDP, length 512
03:33:02.543596 IP 181.21.9.81.33764 > x.x.x.x.54907: UDP, length 512

21:42:04.782150 IP 139.215.201.236.11493 > x.x.x.x.46442: UDP, length 1050
21:42:04.782151 IP 58.251.73.202.33959 > x.x.x.x.32412: UDP, length 1050

09:24:10.500833 IP 178.132.45.135.27844 > x.x.x.x.3074: UDP, length 1024
09:24:10.500933 IP 94.128.247.203.39691 > x.x.x.x.3074: UDP, length 1024

07:34:24.886807 IP 183.165.27.163.6986 > x.x.x.x.80: UDP, length 1400
07:34:24.886807 IP 213.85.158.174.60161 > x.x.x.x.80: UDP, length 1400
```

```
07:34:24.886822 IP 89.113.7.166.63242 > x.x.x.x.80: UDP, length 1400
....E...d...:.e.Yq..H4...
....6[)...g.\....QSq..I..g.;.....S..D&..v.CMI..^_..R}IOP\..#./.K. X.>.rU.W...._:..
fn?U..t.......y|..57..i.....).C..g0....~b.U*....K........+.0..X...Q.;.QEw...o
.....Y.o.N.:....`\Z......&.;...4..b.S. <end packet sample>
```

Figure 3: Attack signatures observed in real-world attacks against Akamai-protected properties.

During these attacks, many of the vectors used were kept with mostly their default values. The obvious exception would be the GET flood, which requires at least some kind of domain to be set. The UDP flood also appears to receive some modification on the default port and packet length. The ACK flood has been one of the most commonly observed vectors in attacks.

/ Attack Signatures Lab/ Mirai comes with an array of attack options along with customizable parameters that allow modification of attack durations, ports, and payloads, to name a few. While the previously provided signatures were observed in attacks against Akamai's customer base, the next series of signatures were acquired during lab testing. While all working attack types are provided, the amount of customization possibilities available for each of these attack types would make it difficult to list all of the attack combinations.

In the next table, all of the attacks found within the source code are listed. In bold are the matching attack names from those observed in attacks on Akamai customers. Some of the attacks, although they may behave slightly differently, are mitigated in a similar way.

```
UDP Flood #define ATK_VEC_UDP          0  /* Straight up UDP flood */
UDP Flood(VSE) #define ATK_VEC_VSE     1  /* Valve Source Engine query flood */
#define ATK_VEC_DNS                    2  /* DNS water torture */
SYN Flood #define ATK_VEC_SYN          3  /* SYN flood with options */
ACK Flood #define ATK_VEC_ACK          4  /* ACK flood */
ACK Flood #define ATK_VEC_STOMP        5  /* ACK flood to bypass mitigation devices */
GRE Protocol Flood #define ATK_VEC_GREIP      6  /* GRE IP flood */
GRE Protocol Flood #define ATK_VEC_GREETH     7  /* GRE Ethernet flood */
//#define ATK_VEC_PROXY                    8  /* Proxy knockback connection */
UDP Flood #define ATK_VEC_UDP_PLAIN    9  /* Plain UDP flood optimized for speed */
GET Flood (can be other request method besides GET) #define ATK_VEC_HTTP   10 /* HTTP
layer 7 flood */
```

Figure 4: List of vectors found in source code. In bold is the vector name as mitigated by Akamai.

Here are the payloads generated within a lab environment. Although some have many options for customization, the default values below appear similar to those observed in live attack traffic. Some sample signatures with customization are also provided.

```
UDP - "STRAIGHT-UP UDP FLOOD"
01:00:48.755295 IP (tos 0x0, ttl 64, id 10062, offset 0, flags [none],
proto UDP (17), length 540)
   192.168.1.230.41088 > 192.168.1.100.10842: UDP, length 512
01:00:48.755297 IP (tos 0x0, ttl 64, id 52201, offset 0, flags [none],
proto UDP (17), length 540)
   192.168.1.230.28336 > 192.168.1.100.44247: UDP, length 512

UDP PLAIN - "PLAIN UDP FLOOD OPTIMIZED FOR SPEED"
00:26:21.861978 IP (tos 0x0, ttl 64, id 50146, offset 0, flags [DF],
proto UDP (17), length 540)
   192.168.1.230.55762 > 192.168.1.100.55762: UDP, length 512
00:26:21.861980 IP (tos 0x0, ttl 64, id 50093, offset 0, flags [DF],
proto UDP (17), length 540)
   192.168.1.230.55762 > 192.168.1.100.55762: UDP, length 512

UDP CUSTOMIZED PARAMETERS
00:29:55.218915 IP (tos 0x0, ttl 64, id 159, offset 0, flags [DF],
proto UDP (17), length 1428)
   192.168.1.230.8008 > 192.168.1.100.8008: UDP, length 1400

SYN - "SYN FLOOD WITH OPTIONS"
01:58:46.835089 IP (tos 0x0, ttl 64, id 62697, offset 0, flags [DF],
proto TCP (6), length 60)
   192.168.1.230.31019 > 192.168.1.100.32420: Flags [S], cksum 0xccc0 (correct),
   seq 2393398558, win 0, options [mss 1402,sackOK,TS val 2928043202 ecr 0,
   nop,wscale 6], length 0
01:58:46.835091 IP (tos 0x0, ttl 64, id 15125, offset 0, flags [DF],
proto TCP (6), length 60)
   192.168.1.230.2665 > 192.168.1.100.40318: Flags [S], cksum 0xc55c (correct),
   seq 4181412311, win 0, options [mss 1402,sackOK,TS val 2928043202 ecr 0,
   nop,wscale 6], length 0

ACK - "ACK FLOOD"
02:17:25.004538 IP (tos 0x0, ttl 64, id 56419, offset 0, flags [none],
proto TCP (6), length 552)
   192.168.1.230.32689 > 192.168.1.100.43222: Flags [.], cksum 0x2592 (correct),
   seq 3111407350:3111407862, ack 797868127, win 34577, length 512
02:17:25.004539 IP (tos 0x0, ttl 64, id 13968, offset 0, flags [none],
proto TCP (6), length 552)
   192.168.1.230.20156 > 192.168.1.100.20623: Flags [.], cksum 0x7d45 (correct),
   seq 628937203:628937715, ack 1238112693, win 34577, length 512
```

```
GRE IP - "GRE IP FLOOD"
01:12:30.178600 IP (tos 0x0, ttl 64, id 24299, offset 0, flags [DF],
proto GRE (47), length 564)
   192.168.1.230 > 192.168.1.100: GREv0, Flags [none], length 544
      IP (tos 0x0, ttl 64, id 35096, offset 0, flags [DF], proto UDP (17), length 540)
   14.57.18.135.9710 > 240.199.228.162.12866: UDP, length 512
01:12:30.178601 IP (tos 0x0, ttl 64, id 11686, offset 0, flags [DF],
proto GRE (47), length 564)
   192.168.1.230 > 192.168.1.100: GREv0, Flags [none], length 544
      IP (tos 0x0, ttl 64, id 47709, offset 0, flags [DF], proto UDP (17), length 540)
   14.57.18.135.47559 > 95.206.109.239.1591: UDP, length 512

VSE - "VALVE SOURCE-ENGINE QUERY FLOOD"
00:31:06.395432 IP 192.168.1.230.63334 > 192.168.1.100.27015: UDP, length 25
       0x0000:  4500 0035 57ec 0000 4011 9e31 c0a8 01e6  E..5W...@..1....
       0x0010:  c0a8 0164 f766 6987 0021 5b69 ffff ffff  ...d.fi..![i....
       0x0020:  5453 6f75 7263 6520 456e 6769 6e65 2051  TSource.Engine.Q
       0x0030:  7565 7279 00                             uery.

00:31:06.395435 IP (tos 0x0, ttl 64, id 9112, offset 0, flags [none],
proto UDP (17), length 53)
   192.168.1.230.50067 > 192.168.1.100.27015: UDP, length 25
00:31:06.395435 IP (tos 0x0, ttl 64, id 16570, offset 0, flags [none],
proto UDP (17), length 53)
   192.168.1.230.6268 > 192.168.1.100.27015: UDP, length 25

DNS - "DNS WATER TORTURE"
00:40:40.611489 IP (tos 0x0, ttl 64, id 52446, offset 0, flags [none],
proto UDP (17), length 73)
   192.168.1.230.17517 > 192.168.1.122.53:
   36644+ A? m3hk3nr6njv0.the-victim.com. (45)
00:40:40.611490 IP (tos 0x0, ttl 64, id 60934, offset 0, flags [none],
proto UDP (17), length 73)
   192.168.1.230.43103 > 192.168.1.122.53:
   19269+ A? htuhwake2bkg.the-victim.com. (45)

HTTP - "HTTP LAYER 7 FLOOD"
00:51:37.153452 IP (tos 0x0, ttl 64, id 11306, offset 0, flags [DF],
proto TCP (6), length 356)
   192.168.1.230.41293 > 192.168.1.100.80: Flags [P.], cksum 0x9101 (correct),
   seq 1:305, ack 1, win 115, options
[nop,nop,TS val 3470182 ecr 868190], length 304: HTTP, length: 304
      GET / HTTP/1.1
      User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36
      (KHTML, like Gecko) Chrome/52.0.2743.116 Safari/537.36
      Host: the-attacktargetdomain.com
      Connection: keep-alive
      Accept: text/html,application/xhtml+xml,application/xml;q=0.9,
      image/webp,*/*;q=0.8
      Accept-Language: en-US,en;q=0.8

HTTP CUSTOMIZED PARAMETERS
17:35:37.875779 IP (tos 0x0, ttl 64, id 24765, offset 0, flags [DF],
proto TCP (6), length 460)
   192.168.1.230.53828 > 192.168.1.122.8008: Flags [P.], cksum 0x1a21 (correct),
   seq 1:409, ack 1, win 115, options [nop,nop,TS val 469359 ecr 57480038],
```

```
length 408
   HEAD /my/path/here.html HTTP/1.1
   User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36
   (KHTML, like Gecko) Chrome/52.0.2743.116 Safari/537.36
   Host: the-attacktargetdomain.com
   Connection: keep-alive
   Accept: text/html,application/xhtml+xml,application/xml;q=0.9,
   image/webp,*/*;q=0.8
   Accept-Language: en-US,en;q=0.8
   Content-Type: application/x-www-form-urlencoded
   content-length: 17

   my=post&data=here
```

Figure 5: Attack payloads generated during lab testing of Mirai malware.

3.0 / Conclusion / While initial attacks in September from the Mirai botnet were the largest ever mitigated, later attacks appeared to drop off in the intensity of attack traffic observed by Akamai. This seems to agree with the original posting of the code leak, where the alleged Mirai author indicated how tens of thousands of bots were no longer connected to the C2 after the attacks on Krebs. It is not clear if this is due to segmentation of this botnet, possibly in a DDoS-for-hire scenario, or ISP filtering of bot traffic. Regardless of the current state of the Mirai botnet, these attacks were an eye opener as to the current potential for harm from DDoS attacks today. While some organizations may be equipped to handle attacks over 100 Gbps, an attack over 600 Gbps may be of greater concern to a larger set of organizations. It's not so much what Mirai brings in the form of attack vectors — VSE, for example, has been around for years in booter/stressor sites. The problem is when hundreds of thousands of devices are told to send as much traffic as they can to a single target. Mirai has been able to successfully compromise a segment that is severely lacking in security best practices within the world of IoT devices. While it's the first malware known to possess this capability, it may not be the last. At the end of the day, what Mirai really brings to the table is that the code base appears to be well written, publicly available (leaked on Github!), and easily extensible. It's unknown as to what Mirai may bring in the foreseeable future, but it is clear that it has paved the way for other malicious actors as they set out to improve on the blueprint.

/ Index /

Value	Name	Description
0x00	Packet size	Length of attack traffic payload
0x01	Random payload	Randomizes payload data (null padded if set to 0)
0x02	ToS	Type of Service field in IP header
0x03	ID	Identification field in IP header
0x04	TTL	Time To Live IP header
0x05	DF	Sets Don't Fragment bit to "on"
0x06	SPort	Source port of attack traffic
0x07	DPort	Destination port of attack traffic
0x08	Domain	Domain name passed in HTTP Host header and DNS attacks
0x09	DNS ID	Domain name transaction ID
0x0a	CC	Sets TCP congestion control
0x0b	URG	Sets TCP header URG flag
0x0c	ACK	Sets TCP header ACK flag
0x0d	PSH	Sets TCP header PSH flag
0x0e	RST	Sets TCP header RST flag
0x0f	SYN	Sets TCP header SYN flag
0x10	FIN	Sets TCP header FIN flag
0x11	SEQ#	Force TCP sequence number
0x12	ACK#	Force ACK number
0x13	GRE mirror	Makes the encapsulated IP the same as the target IP
0x14	Method	GET/POST/HEAD/etc. method in HTTP request
0x15	Data	HTTP POST data in HTTP request
0x16	Path	HTTP path in HTTP request
0x17	HTTPS	Target is running HTTPS (not actively usable)
0x18	Threads	Number of threads/sockets to be used by bot
0x19	Source*	Spoofs source IP In some cases passing the value 255.255.255.255 will result in the spoofing of a completely randomized source
0x1a	Cookie**	HTTP Cookie header

* some attacks do not spoof properly or as expected.
** not implemented in lab-tested binary or present in source code

Customizable field	Default value	Custom value
Packet size	512	1400
ToS	0	1
TTL	64	123
ID	random	1
DF	off	5
Random payload	random	1 (null padded if 0)
SPort	random	31337
DPort	random	8008
Source	non-spoofed	255.255.255.255 (random)

Fig 1.5 shows customizable fields for the UDP attack type

Customizable field	Default value	Custom value
Packet size	512	1400
Random payload	randomized	1 (null padded if 0)
DPort	randomized	8008

Fig. 1.7 shows customizable fields for UDP PLAIN attack

Customizable field	Default value	Custom value
ToS	0	1
ID	random	1
TTL	64	123
DF	true	5
SPort	random	31337
DPort	random	8008
URG	0	1
ACK	0	1
PSH	0	1
RST	0	1
FIN	0	1
SEQ#	random	1
ACK#	0	1
Source	non-spoofed	255.255.255.255 (random)

Fig. 2.0 shows customizable fields for SYN attack

Customizable field	Default value	Custom value
Packet size	512	1400
Random payload	randomized	1 (randomized)
ToS	0	1
ID	random	1
TTL	64	123
DF	false	5
SPort	random	31337
DPort	random	8008
URG	0	1
ACK	1	1
PSH	0	1
RST	0	1
SYN	0	1
FIN	0	1
SEQ#	1	1
ACK#	1	1
Source	non-spoofed	255.255.255.255 (random)

Fig. 2.2 shows customizable fields for ACK attack

Customizable field	Default value	Custom value
Packet size	512	1400
Random payload	random	1 (random)
ToS	0	1
ID	random	1
TTL	64	123
DF	true	5
SPort	random	31337
DPort	random	8008
GRE mirror	false	1
Source	non-spoofed	255.255.255.255 (random)

Fig. 2.4 shows customizable fields for GRE IP attack

Customizable field	Default value	Custom value
ToS	0	1
ID	random	1
TTL	64	123
DF	false	5
SPort	random	31337
DPort	53	8008
Domain	(user supplied)	the-victim.com
DNS ID	random	1

Fig. 2.6 shows customizable fields for DNS attack

Customizable field	Default value	Custom value
ToS	0	1
ID	random	1
TTL	64	123
DF	false	5
SPort	random	31337
DPort	27015	8008

Fig. 2.9 shows customizable fields for VSE attack

Value	Name	Description
DPort	80	8008
Domain	the-victim.com (required & user supplied)	the-victim.com
Method	GET	HEAD
Data	(empty & unused)	my=post&data=here
Path	/	/my/path/here.html
Threads	1	10
Cookie	N/A	N/A

Fig. 3.1 shows customizable fields for HTTP attack

4.0 / DDoS Attack Payloads / Many flood types can be generated using this malware. Figures 4 – 10 show the most common attack types and their respective characteristics

The UDP Flood is generic but allows the control over the payload size and content by the operator as shown in Figures 4 and 5. It can be purposely crafted for bandwidth exhaustion attacks by setting a large payload size or alternatively can be aimed at resource exhaustion with no payload but a high packet-per-second threshold.

```
13:56:46.308717 IP 235.54.204.5.55520 > 127.0.0.1.80: UDP, bad length 24752 > 1472
13:56:46.308726 IP 91.247.24.42.47267 > 127.0.0.1.80: UDP, bad length 24752 > 1472
13:56:46.308736 IP 39.196.144.4.45652 > 127.0.0.1.80: UDP, bad length 24752 > 1472
13:56:46.308745 IP 63.176.1.37.36773 > 127.0.0.1.80: UDP, bad length 24752 > 1472
```